
MASSTOR
SYSTEMS
MASSNET working

IN REVISION

MASSNET

NETWORK
COMMUNICATIONS
SYSTEM

MASSNET
CONCEPTS AND FACILITIES

MASSTOR SYSTEMS CORPORATION

EDITION NOTICE

Revision F, August 15, 1983

Produced: September 6, 1983, 10:03

This is edition F of MASSTOR publication number MSC-1001. In the event of editorial or technical changes, new editions of this document will be issued. The new editions will indicate the appropriate edition designation.

PREFACE

This document provides an overview of MASSNET and describes the facilities available to users of the system. It is a prerequisite for all other MASSNET documentation.

Chapter 1 contains general information. Chapter 2 contains detailed information about the nature of the actual message transmission capabilities and should be read by those who are designing or programming a system using MASSNET.

MASSNET is a complete networking facility that provides a broad range of capabilities between interconnected host computers of the same or different manufacture. Networking with MASSNET provides a computing environment where special computing needs can be addressed, computing resources efficiently shared, and the useful life of presently installed equipment extended.

MASSNET is based on MASSTOR Systems Corporation proprietary software and Network Systems Corporation adapter hardware. Together, these provide the following capabilities:

- Multi-host, multi-vendor capability
- Up to 64 computers can be interconnected at distances of up to 600 feet or 4 computers at over 4000 feet
- High-speed transmission equal to CPU channel data rates
- Easy to use, programmable communications with a choice of two interfaces
- Multi-block chaining, which permits multiple requests to be sent with a single response required
- Remote job/process start
- Communication between network applications residing within the same computer
- Global network management that monitors network status, collects statistics on network utilization, and interfaces to operations
- Global network recovery that provides automatic recovery of connections between the Network Administrator and the various Network Managers after a host computer failure

- Automatic error detection and recovery
- Recoverable connect options that allow applications to recover from host failures
- Global network security that ensures requested connections between applications are permitted

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

MASSTOR Networking Facility (MASSNET) is a complete networking facility that provides a broad range of capabilities between interconnected host computers of the same or different manufacture. Up to 64 computers may be interconnected at distances of up to 600 feet or 4 computers at over 4000 feet.

Figure 1 is a high-level diagram of a representative MASSNET system. Host computers are interconnected using Network Systems Corporation's HYPERchannel, which provides transmission speeds equal to CPU channel rates. MASSNET software in each of the computers is functionally standardized and forms a framework by which independent host computers are integrated into a common network architecture. One of the computers contains additional MASSNET software which provides administrative control of the network.

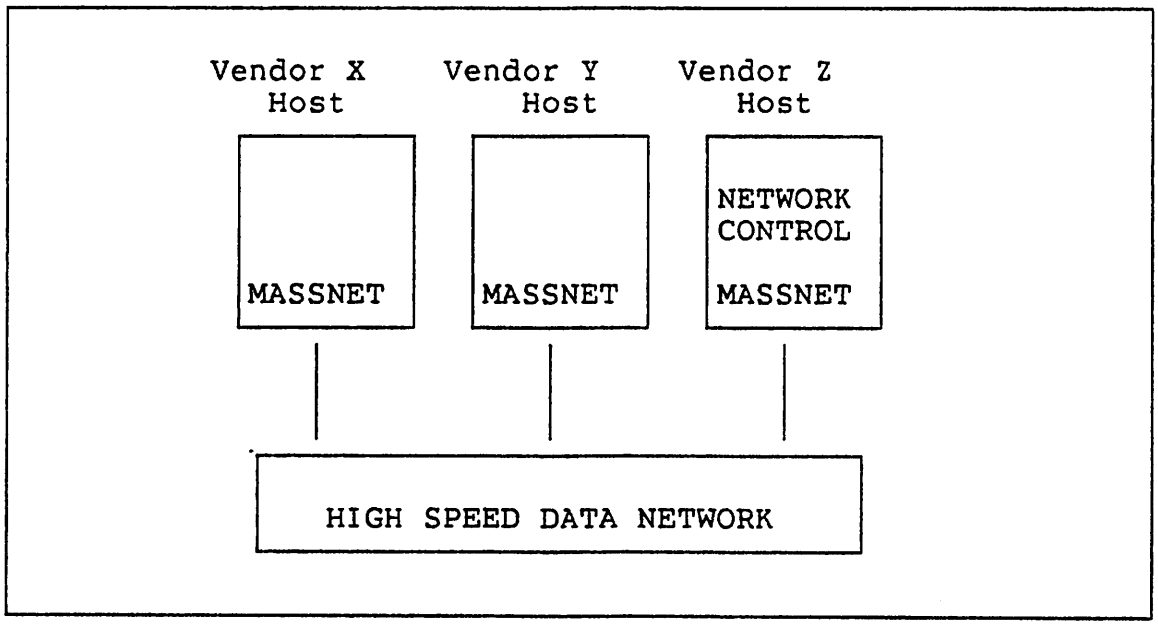

Figure 1: Logical Structure of MASSNET

MASSNET provides both centralized administrative control of the network and the ability to communicate between authorized application programs executing on host computers

connected to the network. This communication can take many forms - messages, records, files - anything the network applications agree to transfer.

The network communication between application programs takes place in parallel; a program in one computer "sends" a message, and the target of this message in another computer must issue a corresponding "receive." Multiple logical connections can take place concurrently over the same adapter hardware.

Network access is made available with easy-to-use subroutine calls through a standardized interface. Communicating application programs use symbolic references (instead of physical hardware addresses) when using MASSNET. Thus, these programs may be moved to different host computers with no change to the program, greatly facilitating operations.

A very useful feature of MASSNET permits a connection between two programs executing on the same computer. Programs need not be aware that communication is being simulated (memory to memory) by MASSNET with no actual network transmission taking place over the adapter hardware. This aids in both program development and testing by eliminating the need for redundant network hardware on each computer.

The network control structure also includes the ability to cause a job or process in one host computer to be started by a job or process executing in another host computer.

No user has direct access to the network hardware and control layers. Therefore, network integrity is ensured; and the user is insulated from device level programming considerations such as interrupt processing, retransmission for automatic error detection and correction, network configurations, and traffic control.

Network management, achieved through the use of a programmed administrator, provides network security, management of network resources, automatic recovery, network auditing, and independence among network connnected computers.

1.1 OPERATION

MASSNET is started by the operators of the various host computers having access to the network. Operator commands initialize the MASSNET software in each computer and start the administrative control of the network. Once this has been done, application programs in the host computers request a network connection through the use of subroutine

calls to MASSNET software. If the request is valid and authorized, MASSNET sets up a logical connection between the sending and receiving application programs. The programs can now send and/or receive data using the MASSNET subroutine interface until they request that the connection be terminated.

The MASSNET software maintains a database of allowable users, connections, security and password requirements, configuration information, current network activity, accounting information, and all data needed to recover from a failure. Console commands allow authorized operators to inquire about the network status and to modify the configuration without disrupting service.

1.2 STANDARD SOFTWARE ELEMENTS OF MASSNET

In each host computer, MASSNET software is installed to access and manage the network. Although the coding is unique for each computer type, the functions are standardized, thus providing uniform interfaces and protocols across a network comprised of unlike computers.

MASSNET is composed of four standard functional software components to handle network management and access:

- NETWORK MANAGEMENT
  - Network Administrator (NETADM)
  - Network Manager (NETMGR)
- NETWORK ACCESS
  - Network I/O Control (NETIO)
  - Network Language Communications (NETCOM)

Figure 2 shows the program content of one of the computers connected to a MASSNET system.

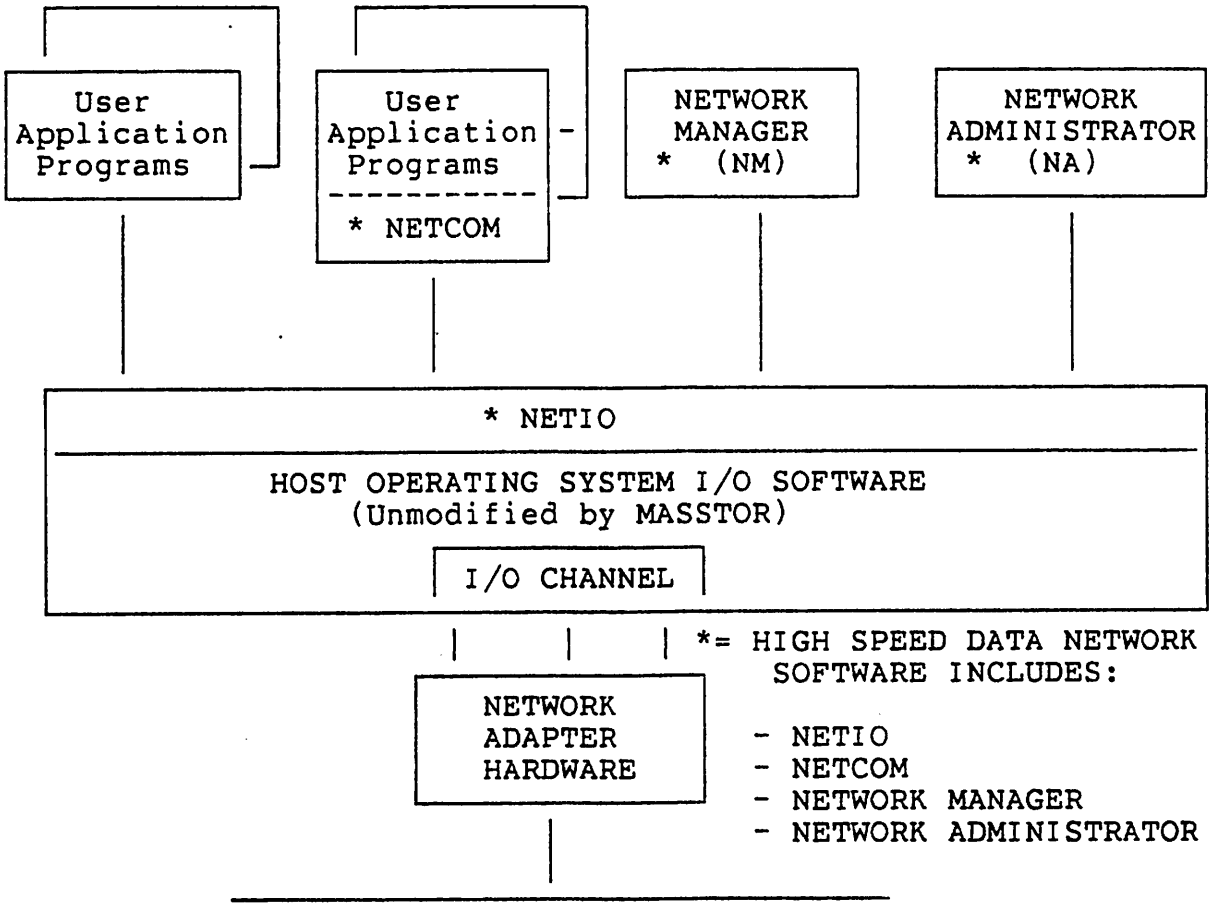

Figure 2: MASSNET Software.

The following discusses each of the MASSNET components at a general level.

1.2.1 Network Management

1.2.1.1 Network Administrator

The Network Administrator (NETADM) is a never-ending application program which executes on one host computer running the IBM MVS operating system in the network complex. Its primary function is to assist local Network Managers (programs executing on each host computer) in establishing and terminating network connections. It also maintains global network configuration information and is the network computer operator's main point of control.

Although active connections are independent of NETADM, no new connections can be made without clearance by NETADM. Therefore, for reliability, NETADM updates a disk database prior to acknowledging any transaction. In this way, network status remains non-volatile and NETADM can be restarted following a system failure without affecting active connections. NETADM usually executes on one of a pair of loosely-coupled computers with shared disks. If the computer running NETADM should fail for any significant period, NETADM may be restarted on the other computer with only minimal delay to pending connection requests.

The primary parameters of a connection request are:

- Topic of conversation
- My application name
- His application name (optional)
- His host name (optional)
- My host name (implied in any connect request)

All corresponding parameters of two requests must match, in order for NETADM to recognize and complete a valid connection. Certain connect parameters can be specified as "don't care". A "don't care" value matches any corresponding parameter except, in certain cases, another "don't care". In addition, NETADM maintains a security file of password data and a user exit interface for a site connection security exit. Each connect request must meet all password requirements on file for its host and application name and must successfully pass the sites security connection exit.

In order to support connect and terminate requests, NETADM keeps track of the overall network configuration with the names of all hosts that are currently active, the network applications currently running on each host, and the network hardware status. NETADM also keeps track of all pending connect requests and supervises the re-establishment of "recoverable connections" after failure of one or both hosts. It is an active task, started by the operator, and can be queried for status, or commanded to perform certain control functions, at any time. NETADM maintains statistics on network utilization and stores these via the MVS System Management Facility (SMF) to provide the means for network application accountability.

1.2.1.2 Network Manager

Network Managers (NETMGR) are never-ending application programs which execute on each computer in the network complex. This is an active process, started and controlled by the computer system operator. The primary functions of the NETMGR are to monitor the local configuration and to establish and terminate (via NETADM) connections on behalf of processes executing on its computer. Each NETMGR maintains local configuration data and can communicate with the local console operator.

A NETMGR also communicates with NETADM across a special, private connection established when NETMGR is initialized. A low-level "handshaking" exchange takes place continuously across this connection, even when NETMGR has no request to make of NETADM. This handshaking informs NETADM that his hosts are up and running, allows each NETMGR to monitor the status of NETADM, and performs a continuing diagnostic check of the network hardware. It also keeps NETADM current about the status of each NETMGR.

1.2.2 Network Access

1.2.2.1 Network I/O Control

Network I/O Control (NETIO) is the access method of MASSNET. It interfaces all user programs to the network regardless of whether a higher-level subroutine interface is used or NETIO is called directly. NETIO has responsibility for overall control of the networking hardware, including interrupt handling, exception processing, and request queueing. Therefore, the NETIO caller need not be concerned with network adapter capacities, traffic levels, or trunk and node configurations.

The following is a partial list of the user-callable functions of NETIO:

- Acquire (Open)

  Dynamically allocate and initialize a network logical unit.

- Connect

  Establish a connection with a program on a remote (or local) host computer through a logical connection. A connect operation can specify that the remote application program is to be automatically started on the specified remote computer.

- Send

  Transmit a message (and optionally associated data) across a specific connection. A send request operation includes an automatic receive response except in a multi-block start of chain or middle of chain operation.

- Receive

  Receive a message (and optionally associated data) from one specific connection or any one of a group of connections on a logical unit.

- Terminate

  Terminate a connection.

- Release

  Release a logical unit.

The implementation of NETIO will vary among computers and operating systems. Although the logical functions are the same, each implementation must fit into the structure of its host executive and I/O system. NETIO is intended for use by assembly language programs and follows the programming conventions of each local host assembler and operating system.

A powerful feature of NETIO is the ability to establish connections between two programs running on the same computer. NETIO handles this at a low level so that programs need not be aware that local communication is being simulated by NETIO with no real network I/O actually taking place.

1.2.2.2 Network Communications Language

The Network Communications Language (NETCOM), an optional component of MASSNET, is a high-level language 'CALL' interface to the facilities of NETIO. NETCOM eliminates system dependent programming considerations and makes network communications even simpler. It may be used by programs written in COBOL, FORTRAN, PL/1, Assembly Language, and others. It provides a high degree of program transportability from one computer type to another.

NETCOM consists of a set of subroutines which enable user programs to interface with the facilities of the network.

These subroutines can be invoked by programs written in PL/I, Fortran, Cobol, and Assembly language. Their names and functions are:

NCONN	-	establish a network connection
NSEND	-	send message and data
NRECV	-	receive message and data
NSENDR	-	send followed by receive message and data
NTERM	-	terminate a network connection
NWAIT	-	wait for completion of user-synchronized I/O
NERROR	-	return error information and text

The following example illustrates what an application programmer would code in his program using NETCOM to establish and terminate a network connection, send a message, and receive a response from his network partner.

```
----
----
CALL NCONN        (Acquires resources and
----               establishes connection)
----
CALL NSENDR       (Sends message and receives
----               response)
----
CALL NTERM        (Terminates connection and
----               frees resources)
```

1.3 OPTIONAL SOFTWARE ELEMENTS OF MASSNET

1.3.1 MASSLINK

MASSNET also has as an optional software feature used to extend the scope of the network beyond the "local" site:

- MASSLINK
  - Linked host configuration and control
  - Link adapter support

MASSLINK expands the capabilities of MASSNET by extending the high speed data network (HSDN) to distances of up to hundreds of miles. MASSLINK, through the use of link adapter hardware, allows host computers using MASSNET to communicate with each other over various type of land line and microwave systems (both leased and private). The method of communication between the link adapter hardware, as supported by MASSLINK, is in half-duplex mode over a

full-duplex line connected using standard data communications equipment (DCE) interfaces. Some examples of the supported interfaces are RS232-C, Bell 303, and Bell 306. MASSLINK communications will support any line speed used between the DCE connections from 44 bits to 44.7 megabits per second.

The primary objective of the MASSLINK software is to permit any MASSNET application on a MASSNET host computer to access any other MASSNET application on any other host computer, within the confines of NETADM security, without regards to the physical distances involved. It is assumed that they are connected together via link adapters which are available on the trunk to the host computers processor adapters.

Both application partners and MASSNETs will be considered to be all in the same "MASSNET network" (that is one NETADM will be controlling the network). The hosts in such an environment are considered to be "linked" network hosts. Also the connections established between them are "linked" network connections, and are treated as such. MASSLINK will handle communications over "n" number of link adapters (also known as multi-hop capability) as long as the links are correctly placed in the network in a valid link adapter hardware configuration, and with the correct group table address settings.

From a user's viewpoint the optional MASSLINK facility in a MASSNET environment is invisible. All applications that will run with standard MASSNET will run in an environment with MASSLINK linked hosts without modification, as regards the use of MASSNET and its interface. Any internal timing dependancies of the application program pairs may have to be adjusted to take into account the lower speed of some DCE link connections. With MASSLINK, normal operational procedures of the MASSNET environment continue, as is described in the various host computers MASSNET operators' manuals, except where changes are required to support the linked hosts.

Figure 3 shows two of the various combinations of MASSNET linked host computers using MASSLINK.

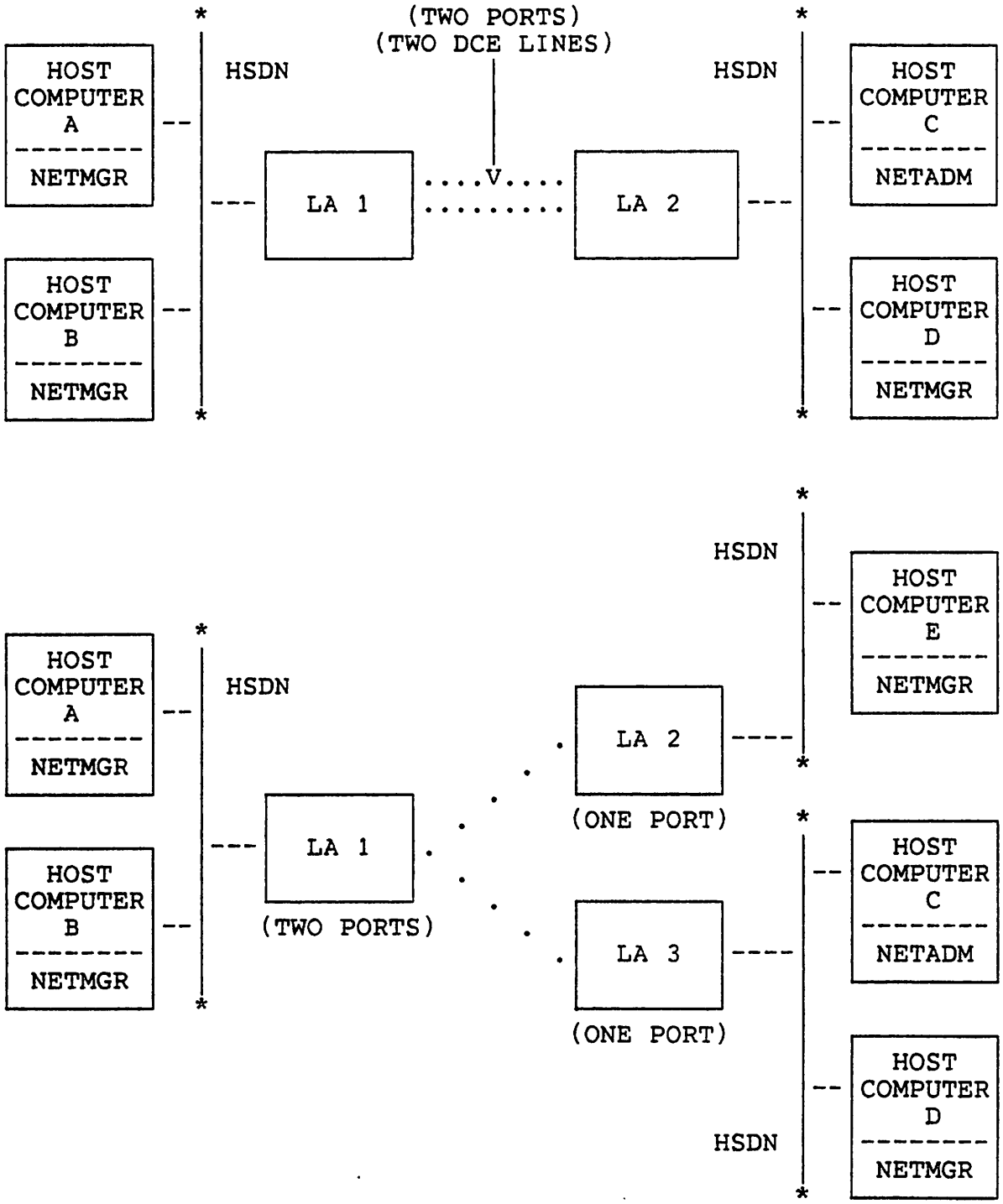

Figure 3: Typical MASSLINK Link Adapter (LA) Configurations

1.4 FLOW OF CONTROL

Figure 4 illustrates the typical sequence of events in establishing a network connection. The numbers represent one sequence in which these events could take place. In any order, user programs on Hosts A and B issue CONNECT requests via NETIO. Within each host, NETIO registers the CONNECT request with the local NETMGR which, in turn, forwards this request via NETIO to the Network Administrator, wherever it may be. When NETADM has received both requests and determined that they are valid and matching, it sends connection complete messages to both NETMGRs. These connection complete messages include the network hardware and NETIO software address of the other application program - plus accounting and security information used by each NETMGR on the cooperating host to complete the internal data structures corresponding to the new connection. These data structures are shared by the NETMGR task and NETIO operating on behalf of the user task. NETMGR posts NETIO that the connection is complete and NETIO posts the user. From now on, user I/O requests (SEND, RECEIVE) are processed directly by NETIO without the assistance of NETMGR or NETADM.

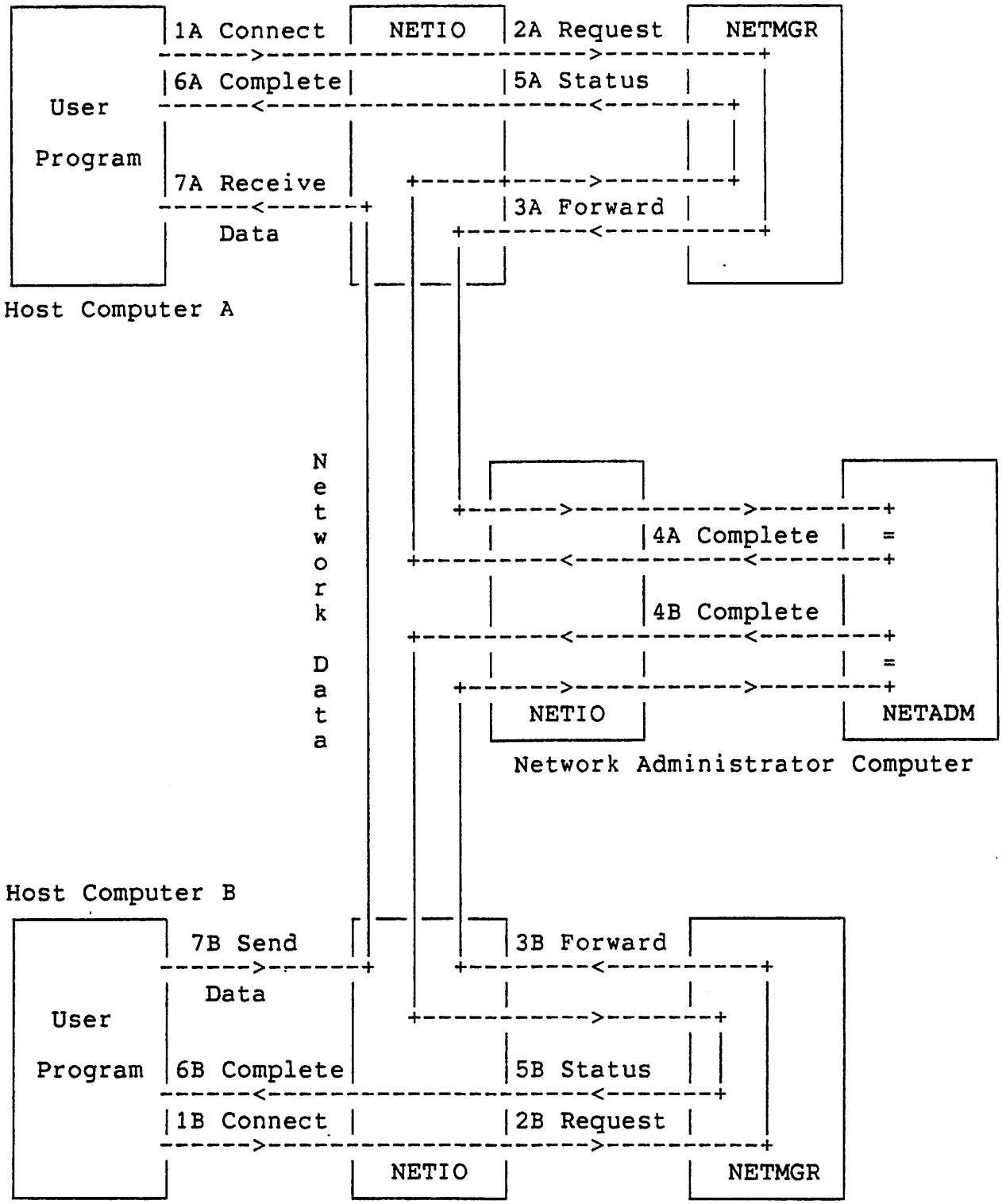

Figure 4: Network Connect Protocol

2. NETWORK MESSAGES

2.1 NETWORK TRANSMISSIONS

All network transmissions consist of two parts as shown in Figure 5.

- A fixed-length 64-byte "message proper" and
- An optional variable-length block of "associated data".

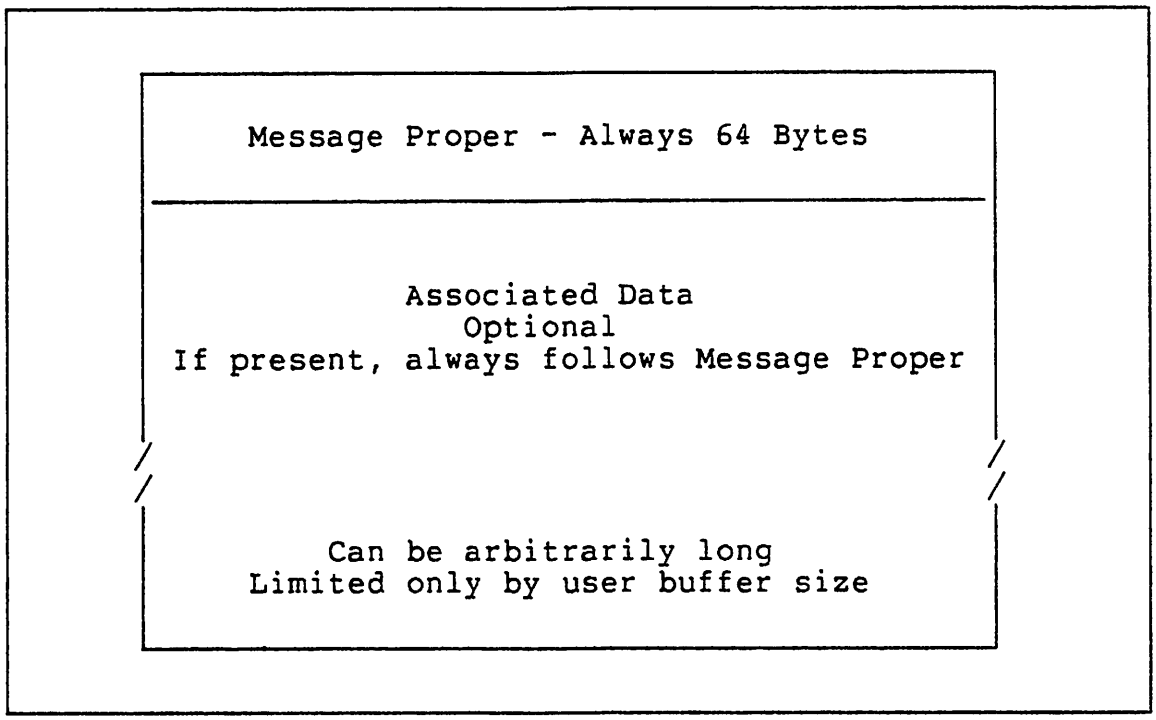

Figure 5: Network Transmissions

From a programmer's viewpoint (and also as seen by the host computer channels), the message proper and its optional associated data block are a tightly-coupled entity. That is, the message proper and its associated data belong together and are never intermixed with other unrelated fragments of message and/or data. A similar analogy is that of the count field and data field on an IBM disk drive. The network adapter hardware, however, multiplexes bursts of messages and data on the I/O trunks with other data from other adapters. The messages and associated data are repackaged by the receiving adapters such that host computer channels and host programs see only correlated streams of message and associated data.

2.1.1 Message Proper

The 64-byte message proper is used primarily by the network adapter hardware and NETIO software to route messages (and associated data) between cooperating host application programs. It can also be used to transmit small amounts of application program data.

The message proper is divided into three parts as shown in Figure 6 below.

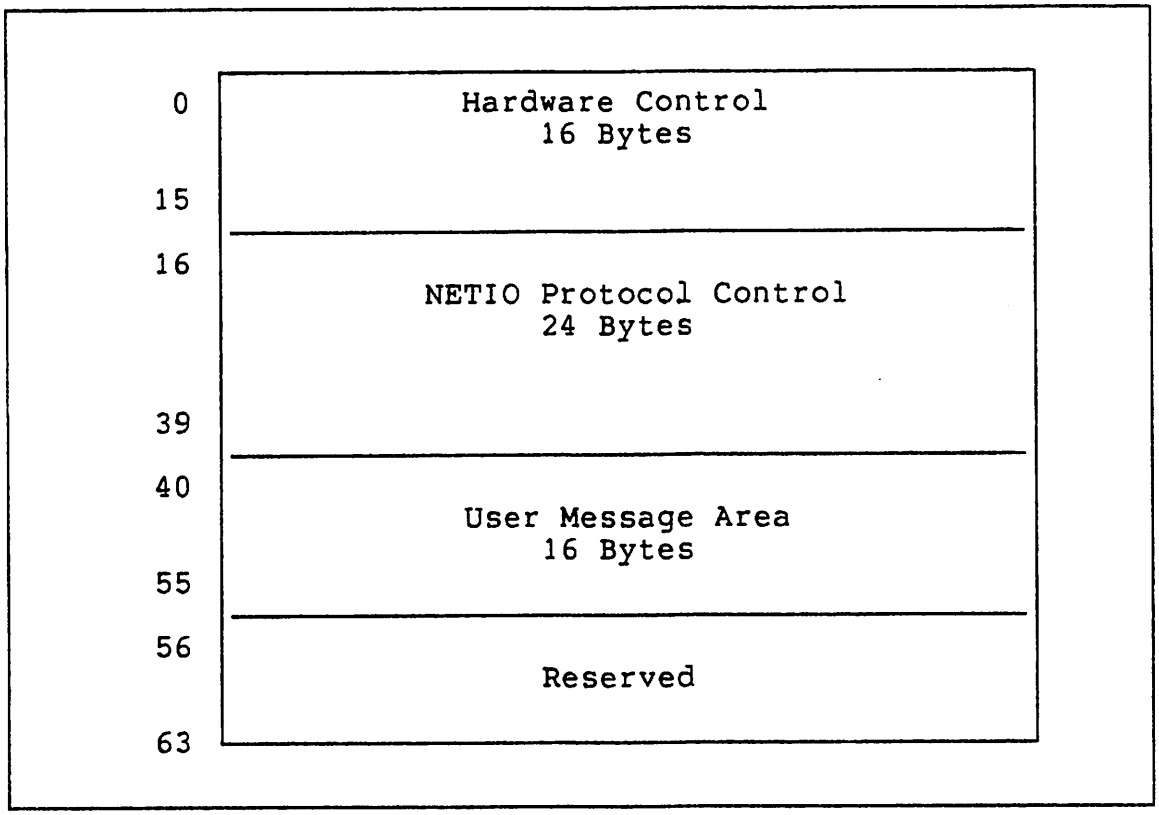

Figure 6: Network Message Proper

- Hardware Control

  The first 16 bytes of the message proper (bytes 0-15) are used by the network adapters to select an I/O trunk, to specify the physical unit address of a remote network adapter and its security access code, and (on some adapters) to indicate the destination host logical unit address for the message. A bit in this control portion also indicates to the adapter whether or not associated data is present with the message. The hardware control bytes of the message proper are completely managed by NETIO control

software and cannot be directly examined or modified by application software in any way.

- NETIO Protocol Control

  The next 24 bytes of the message proper (bytes 16-39) are used by the NETIO control software to manage and enforce network protocol. The protocol control bytes in the message proper are completely managed by NETIO control software and cannot be directly examined or modified by application software. This is an integrity feature of MASSNET that precludes incorrectly functioning user applications from propagating their failure to the rest of the network. These protocol fields include:

  - A globally unique network connection number assigned by the Network Administrator for every network connection.
  - Two locally unique local connection numbers assigned by the Network Manager on each host for each side of a network connection.
  - A Request/Response sequence number used to prevent the loss of messages and data.
  - The length in bytes of associated data (if any) used by NETIO to ensure that data fragments are not lost in the network.
  - Request and Response timeout values.
  - Various other protocol and security fields.

- User Message Area

  The next 16 bytes of the message proper (bytes 40-55) are available for application programs to exchange limited amounts of application information. Even these 16 user bytes are not directly addressable by user programs; they are managed by NETIO control software which moves the user message into and out of this area at user program request.

- The last 8 bytes are reserved for future expansion (bytes 56-63).

2.1.2 Associated Data

An optional, arbitrarily long block of user data can be associated with every network message transmission. The associated data is not examined by the network adapter

hardware or by NETIO control software, except to count and verify its correct length as specified by the application program.

Exception: On some adapters, a user can (via NETIO) request the hardware to perform limited format conversion. The details of this coversion are unique to every adapter type and are beyond the scope of this document.

Because host computer channels are relatively slow (about 12 million bits per second) and the network hardware trunks connecting network adapter hardware units are relatively fast (about 50 million bits per second), the network adapter hardware interleaves fragments (frames) of associated data between one pair of adapters with fragments of associated data (and messages) between other adapter pairs. This multiplexing is strictly in the hardware; application programs cannot detect its occurrence and should treat every block of associated data as a single entity associated with its parent message proper.

2.1.3 Network Hardware Protocol & Error Checking

The network adapter hardware has a strict internal protocol regarding I/O trunk access, device selection and reservation, and collision avoidance. User programs are unaware of this and need not be concerned with it.

The network adapter hardware generates and decodes error detection and correction data which is injected into every I/O trunk transmission. It autonomously manages error recovery and retry for transient errors. Detected errors which cannot be corrected by the hardware are retried by NETIO control software. Detected hardware I/O errors which are uncorrectable by adapter hardware or NETIO software are indicated as error status to the user application program(s) which should consider the error permanent, i.e., user-level I/O error recovery/retry is not recommended.

2.2 PROTOCOLS AND INTERFACES

MASSNET is a layered networking system with a separate protocol for communications between corresponding layers at either end of a connection.

Figure 7 illustrates six of the seven protocols in the MASSNET architecture. The seventh protocol regards the establishment of connections prior to application program data exchange over the network and is discussed separately. These protocols prescribe strict adherence to correct etiquette and procedures in exchanges between pairs of components at the SAME level in the network hierarchy, i.e., the horizontal lines in the figure. It is an important concept of the MASSNET architecture that none of the protocol lines cross, i.e., each layer is independent of the layers above and below from a protocol standpoint. There is no mixing of application protocol with network protocol. This enables the easy development of network applications and enables the layering of higher application-to-application protocols without specific regard to lower protocol layers.

The interfaces are the translation mechanisms between components at different levels in the network hierarchy, i.e., the vertical lines in the figure - the compiler or assembler between a user and his application program, the control blocks or parameter lists passed between an application program and NETIO (or NETCOM), the NETIO to channel and channel to adapter interface, etc.

2.2.1 Trunk Hardware Protocol

At the lowest level, the network adapter trunk interface hardware has a protocol for access to the trunk(s) and for correctly delivering frames of data between pairs of adapters. All trunk interface boards are identical in operation. Supported by adapter microcode, they generate and verify error correction and detection data in the transmitted frames. They also perform trunk acquisition arbitration, listen for incoming traffic, and keep track of trunk status.

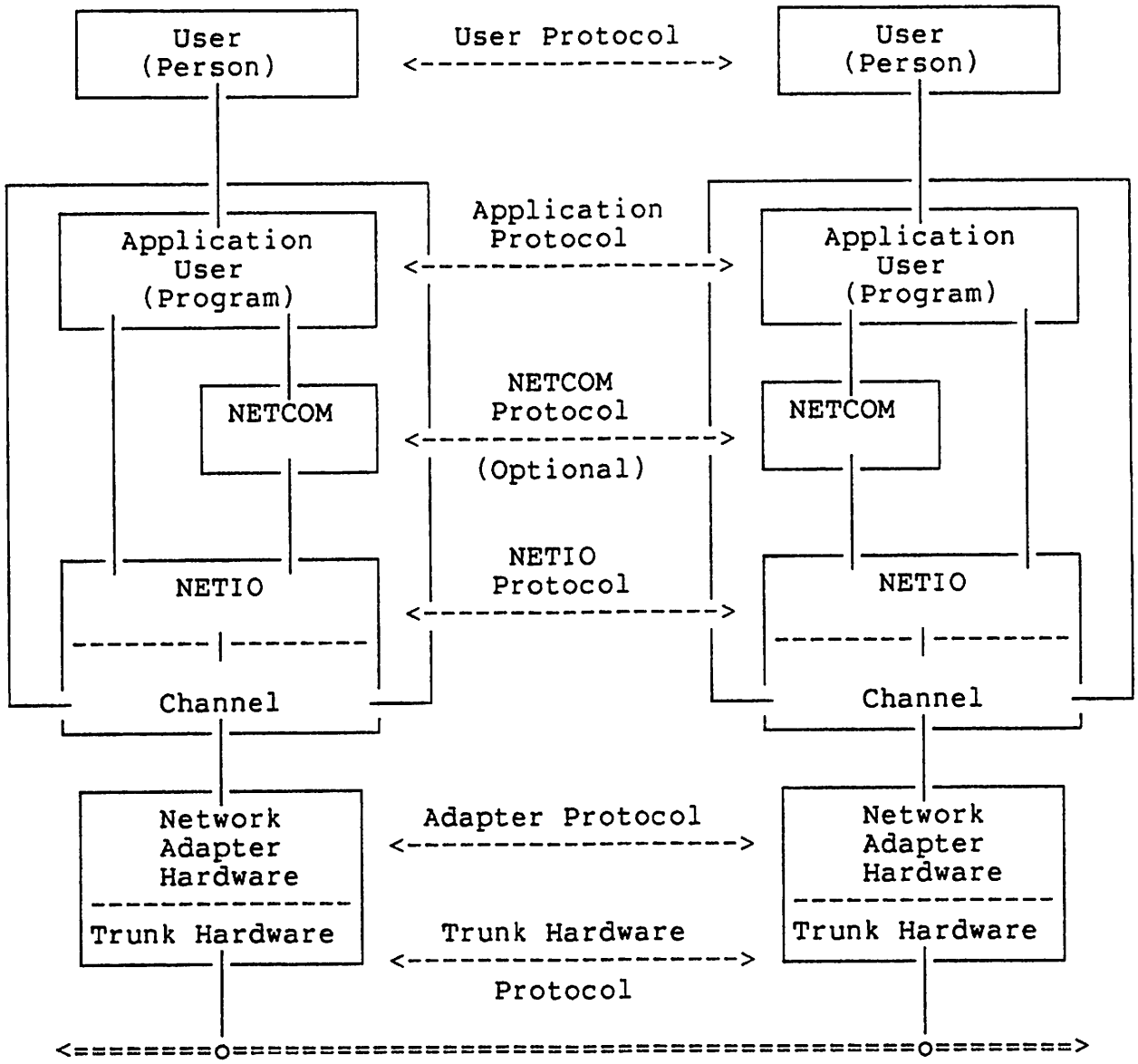

Figure 7: Network Protocol Layers

2.2.2 Network Adapter Hardware and Microcode

The network adapter base hardware receives error-corrected data from the trunk interface boards. The inter-adapter protocol involves the successful transmission of larger amounts of data than the individual trunk frames. The adapters synchronize the transmission, receipt, and retransmission (if necessary) of whole messages and associated data up to 2048 bytes (4096 bytes and larger

frames on some adapters) in a buffer segment. Amounts of associated data larger than individual frames are transmitted by an inter-adapter double buffer protocol. The network adapters respond to commands received on the host channel interface, present data and status, and (on some adapters) generate attention interrupts to the channel interface.

Data which arrives at a host channel interface from a remote adapter has been handled by two lower level protocols - the inter-adapter protocol and the trunk interface board protocol. There is no channel-to-channel protocol.

2.2.3 NETIO Protocol

NETIO is the first level of host-resident network protocol. NETIO is the delivery mechanism for network transmissions (message and associated data). Incoming transmissions are delivered to main storage buffers for the receiving process (application program); outgoing transmissions are delivered to the network from user program buffers. The inter-NETIO protocol is based on exception reporting; that is: successfully delivered transmissions are not specifically acknowledged to a remote NETIO by a local NETIO. Acknowledgement of successfully received messages is the responsibility of the application program. The application program can defer a response until the message has been successfully processed. Thus, there is no NETIO acknowledgement overhead in addition to the application program processing acknowledgement. The protocol exceptions which are reflected to a remote NETIO include:

- Network errors reported on the channel interface
- Receipt of unsolicited transmissions for which there is no user receive active or no connection
- Incorrect transmission sequence number
- Incorrect length of associated data
- Response timeout

NETIO manages the first 16 bytes of the message proper for the adapter and trunk hardware interfaces. NETIO manages the 24-byte protocol control portion of the message proper for the NETIO protocol. The entire message proper is managed by NETIO in protected regions of host computer main storage unavailable for access or modification by application programs.

NETIO is designed to service many local host applications, each carrying on independent conversations with remote

partner applications on different remote computers. The NETIO protocol is a multi-path protocol among all NETIOs on the network on a transmission-by-transmission basis.

2.2.4 NETCOM Protocol

An optional layer of protocol is provided by the NETCOM facility for general-purpose application programs which choose to use NETCOM for programming convenience. In simple terms, NETCOM automatically generates the required network response to a request received from a remote application program which is also using NETCOM. Application programs using NETCOM send and receive only requests over the full duplex connection supported by NETIO. Responses are an automatic, internal function of NETCOM.

An advanced feature of NETCOM intended for linked, independent networks, includes the ability to send multiple requests in a chain with a NETCOM response required only to the last request in a chain.

2.2.5 User Application Program Protocol

User application programmers are free to design their own inter-application program protocols within the rules and constraints of the lower-level MASSNET protocols of NETCOM if used, of NETIO, and of the adapter and trunk hardware. The rules in summary are:

- A proper connection must be established between user application programs via the connect protocol provided by the Network Managers and the Network Administrator.
- Except for multi-block chaining, only one request can be sent on a connection before waiting for a response.
- Except for multi-block chaining, every request received must be acknowledged by a response to the sender within the agreed timeout limit.
- One or more receive request operations can be initiated at any time. These are posted complete as requests arrive (or time out if no requests arrive in the specified time).
- Application programs (via NETIO) can individually specify request and response timeout values.
- Application programs must handle their own inter-application protocol errors. These include:

- Partner terminated
- Partner not receiving
- Timeout
- Incorrect buffer size
- Local protocol violations

2.2.6 User-level (Human) Protocol

At this level of the network protocol hierarchy, program designers are encouraged to use their creativity in employing the powerful facilities of the MASSNET High Speed Data Network. Their only constraints are:

- Consistent application program names and connection protocol must be used in identifying their applications to the Network Managers and the Network Administrator.
- There must be consistent agreement regarding the content, format, size, and meaning of the network user messages and associated data.
- Each user must understand the capabilities and limitations of their own and their partner's host computer and operating system. The MASSNET facilities are as consistent as is technically possible, given the architectural differences between the various host computers.

2.3 LOGICAL REQUEST/RESPONSE PROTOCOL

NETIO control software manages every connection between cooperating application programs such that they appear to have two independent, communication paths - one inbound and one outbound. Figure 8 illustrates the full-duplex connection between a local application program and its remote network partner.

As this figure shows, the outbound link and the inbound link are TOTALLY independent and TOTALLY symmetrical. Both requests and responses can contain associated data in either direction. Both inbound and outbound traffic can flow simultaneously; an outbound request and an inbound request can pass each other without error in the network. Applications which are using asynchronous I/O can wait on multiple events - the arrival of either a request or a response - and the inbound and outbound transmissions can be multiprogrammed in complex applications.

2.3.1 Outbound Requests and Responses

The outbound half of a NETIO connection consists of REQUEST messages (and associated data) sent to a remote application and RESPONSE messages (and associated data) received from a remote application. The notation "SREQ" or "send request" and the corresponding graphic "===>" indicate outbound requests. The notation "RRSP" or "receive response" and the corresponding graphic "<---" indicate a response received on the outbound link (even though the direction of data flow for these "outbound" responses is inward).

Except in multi-block chaining mode, every outbound request sent must be synchronized with a received response before another request can be sent. The NETIO and NETCOM access methods automatically combine a SREQ and RRSP in a single call. It is an application program protocol error to send more than one request without an intervening response.

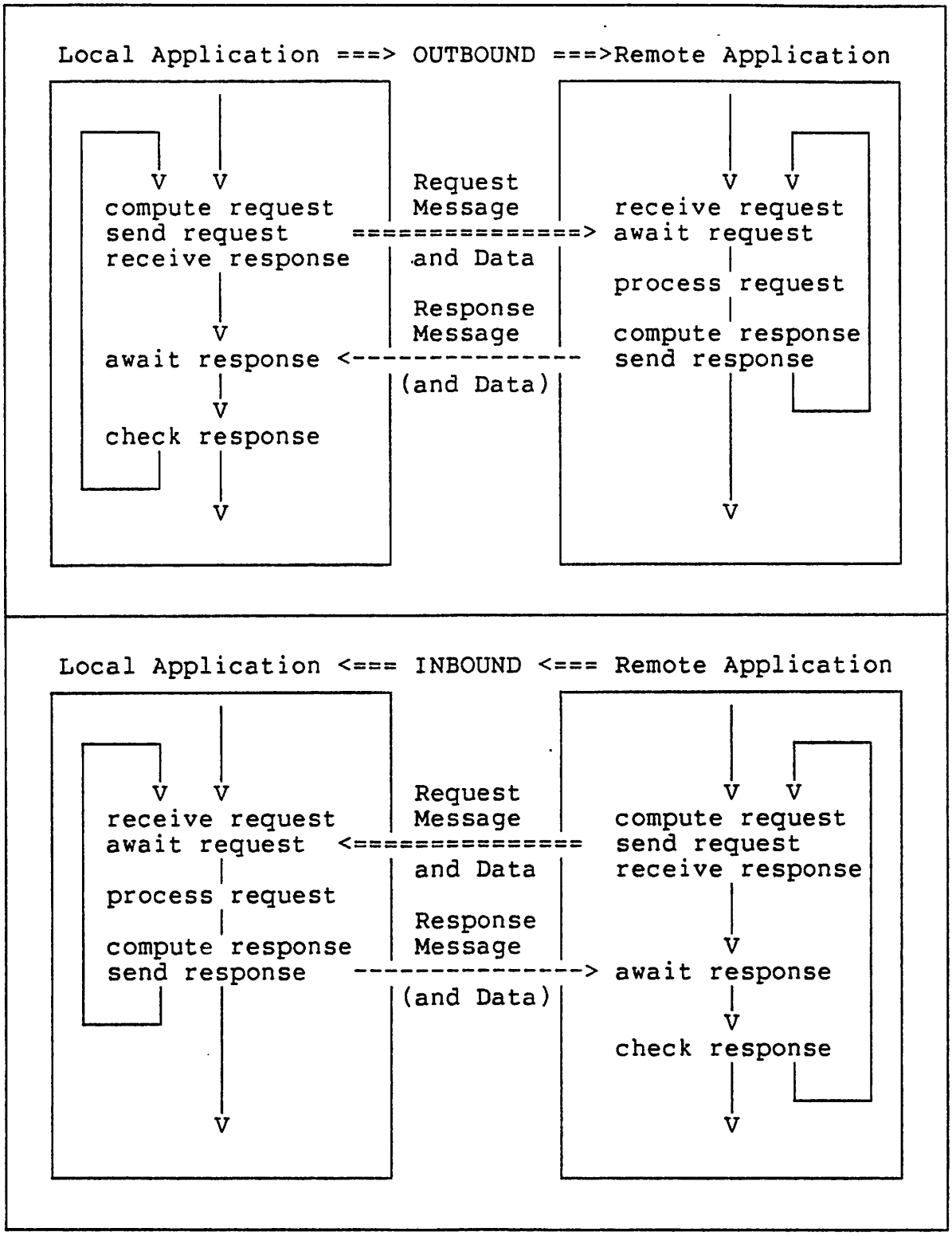

Figure 8: Logical Request/Response Concept

2.3.1.1 Send Request

After validating the syntax and local protocol for the operation, NETIO constructs the outbound message proper including:

- The hardware to/from address and control fields maintained by NETIO in protected storage for this connection.
- The NETIO protocol control fields including:
  - A request sequence number which is incremented by one for every outbound request.
  - A response timeout value (specifiable by the user).
  - The to/from local connection numbers for this connection.
  - The network connection number.
  - The length of the user message and associated data if any.
  - Other NETIO internal data.
- The user's 16-byte message proper moved from user storage into the message proper.

This message (and associated data if any) is transmitted to the remote network adapter. Transmission errors detected at this time are retried automatically by NETIO.

Upon receipt of a request message at the remote host, the remote NETIO performs the following functions:

- Read the message proper into protected NETIO storage.
- Search the connect table for the local connection number.
- Verify the network connection number.
- Verify the request sequence number for 1+ the last request received.
- Verify user has receive request pending.

- Verify user input buffers are available.
- Move 16-byte user message to user's buffer if requested.
- Read associated data (if any) into user's buffer.
- Post the user with the status of its receive request operation.
- Start the clock running on the response time value specified by the sending program.

When the send request operation completes (normally), the sending NETIO starts a timer which will cause the request to be retransmitted if the remote application does not send a response before this time has expired.

2.3.1.2 Receive Response

Except in multi-block chaining mode, a receive response operation (RRSP) is automatically associated with an outbound send request. RRSP never causes any network I/O to be initiated; it is queued internally by NETIO in anticipation of receiving a response message (and data) to a request previously sent. The response can be generated by the remote application or by the remote NETIO (a NETIO generated NAK).

Upon receipt of a response message, NETIO performs the following functions:

- Read the message proper into protected NETIO storage.
- Search the connect table for the local connection number.
- Verify the network connection number.
- Verify the response sequence number for equality with the last request sent.
- Verify user has receive response pending.
- Move the 16-byte response message to user's buffer.
- Read the associated data (if any) into user's buffer.
- Post user with status of receive response operation.

When the RRSP operation completes (normally), the receiving NETIO posts completion status to the receiving program. The remote NETIO posts completion status to the program which sent the response as soon as the transmission was complete (not necessarilly received by the local application program). There is no response to a response.

2.3.2 Inbound Requests and Responses

The inbound half of a NETIO connection consists of REQUEST messages (and associated data) received from a remote application and RESPONSE messages (and associated data) sent to a remote application. The notation "RREQ" or "receive request" and the corresponding graphic "<===" indicate an inbound request. The notation "SRSP" or "send response" and the corresponding graphic "--->" indicate a response sent on the inbound link (even though the direction of data flow for these "inbound" responses is outward).

Except in multi-block chaining mode, every inbound request received must be acknowledged with a response message sent to the remote application. It is a protocol error to send a response for which there has been no request received. It is also a protocol error to fail to send a response to a received request within the agreed response time.

Unlike send request, which is automatically associated with a receive response (by NETIO or by NETCOM), receive request and send response are separate functions.

2.3.2.1 Receive Request

A receive request operation (RREQ) never causes any network I/O to be initiated; it is queued internally by NETIO in anticipation of receiving a request message (and associated data) from the remote application. After queueing the RREQ operation, NETIO starts a timer (the value of which is specifiable by the user) which will generate an error return code if a request is not received within this time.

Upon receipt of a request message, NETIO performs the following functions:

- Read the message proper into protected NETIO storage.
- Search the connect table for the local connection number.
- Verify the network connection number.

- Verify the request sequence number for 1+ the last request received.
- Verify user has receive request pending.
- Move the 16-byte request message to user's buffer.
- Read the associated data (if any) into user's buffer.
- Post user with status of receive request operation.

When the RREQ operation completes (normally), the receiving NETIO posts completion status to the receiving program. The remote NETIO which sent the request automatically queues a receive response (and starts a timer for the awaited response).

2.3.2.2 Send Response

After validating the syntax and local protocol for the operation; NETIO constructs the outbound message proper including:

- The hardware to/from address and control fields maintained by NETIO in protected storage for this connection.
- The NETIO protocol control fields including:
  - A response sequence number which is the same as the last received request to which this is a response.
  - The to/from local connection numbers for this connection.
  - The network connection number.
  - The length of associated data if any.
  - Other NETIO internal data.
- The user's 16-byte message proper moved from user storage into the message proper.

This message (and associated data, if any) is transmitted to the remote network adapter. Transmission errors detected at this time are retried automatically by NETIO.

When the SRSP operation completes, the sending NETIO posts completion status to the sending program (which presumably will issue another RREQ). The remote NETIO will receive the response and post completion status to the remote program (which presumably will issue another SREQ). There is no response to a response.

This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate.

Do not use this form to order publications; they may be ordered via your MASSTOR representative, the branch office serving your area, or from the MASSTOR address on the reverse side of this form.

How is this publication used ? (Check those which apply)

[] As an introduction

[] As a text (student)

[] As a reference manual

[] As a text (instructor)

[] For another purpose (please explain):

_____

_____

Thank you for your cooperation.
If you wish a reply, give your name and address:

_____

_____

_____

READER'S COMMENT PAGE

Fold here

FROM:

TO:
MASSTOR SYSTEMS CORPORATION
541 Lakeside Drive
Sunnyvale, California 94086-9803

ATTN: DOCUMENT CONTROL

Fold here

READER'S COMMENT PAGE